CLAIM AMENDMENTS

Please cancel claims 2-3, 5, 9, 10, 13, 17, 21, 24, and 26-31 without prejudice or disclaimer.

Please amend claims 1, 4, 7, 8, 11, 12, 14, 15, 18, 19, and 25 as follows.

1. (Currently Amended) A method, comprising:

storing a main software thread in a first hardware context;

binding a first speculative thread to a second hardware context and a second speculative thread to a third hardware context;

dynamically invoking **[[a]]** the first speculative thread ~~from a~~ using the main thread ~~in a processor;~~

dynamically invoking the second speculative thread using the first speculative thread;

transferring live-in values from the main thread to the first speculative thread;

transferring live-in values from the first speculative thread to the second speculative thread;

tracking for a set of loads of interest a number speculative threads that have been spawned relative to a number of instances of any load of interest that have not yet been retired by the main thread; and

decrementing a counter when the main thread retires the corresponding load of interest

~~executing instructions comprising the speculative thread~~.

2. (Canceled).

3. (Canceled).

4. (Currently Amended) The method of claim **[[3]]** 1, further comprising raising an exception and branching to a recovery handler ~~when it is determined that the attempt to bind a hardware thread context for the speculative thread was a success~~.

5. (Canceled).

6. (Original) The method of claim 1, further comprising transferring live-in values for pre-computation slices to hardware thread context register files.

7. (Currently Amended) The method of claim 6, further comprising:

allocating a portion of on-chip memory buffer space for the first and second speculative threads; and

dedicating the allocated a portion of on-chip memory buffer space as an intermediate buffer to pass live-in values from the main thread to the first speculative thread and from the first speculative thread to the second speculative thread.

8. (Currently Amended) The method of claim 6, further comprising:

identifying at least one load of interest;

constructing pre-computation slices for each load of interest, wherein the pre-computation slice comprises a speculative thread that pre-computes an address accessed by a load of interest and pre-fetches ~~fetching~~ for the load of interest using the pre-computed address; and

establishing triggers to invoke the speculative thread.

9. (Canceled).

10. (Canceled).

11. (Currently Amended) The method of claim [[9]] 1, further comprising:

detecting a trigger to invoke the second speculative thread[[,]];

storing second speculative thread live-in values to a buffer[[,]]; and

executing instructions in the second speculative thread.

12. (Currently Amended) The method of claim [[10]] 8 wherein [[each]] an individual speculative thread includes a pre-computation slice, the method further comprising:

allocating an entry in a queue for a copy of at least one pre-computation slice when it is determined that a hardware context is unavailable for the copy of the pre-computation slice; and

placing the pre-computation slice in the queue until a hardware context is available.

13. (Canceled).

14. (Currently Amended) The ~~processor~~ method of claim [[13]] 1, further comprising transferring ~~wherein the logic to copy~~ live-in values from ~~first hardware context to the second hardware context includes~~ flash-copy hardware or a portion of memory buffer space.

15. (Currently Amended) An apparatus ~~processor~~, comprising:

a processor having:

a first hardware context to store a main thread;

a second hardware context and a third hardware context to bind to a first speculative thread and a second speculative thread, respectively, the main thread to dynamically invoke the first speculative thread and the first speculative thread to dynamically invoke the second speculative thread; [[and]]

a processor-readable medium having at least one processor-readable instruction stored thereon to instruct the processor to trigger the invocation of the first and second speculative threads;

logic coupled between the first, second, and third hardware contexts, and [[a]] the processor-readable medium ~~having processor-readable instructions stored thereon to instruct the processor~~, the logic to bind the first and second speculative threads to the second and third hardware contexts, respectively, and to transfer live-in values from main thread to the first speculative thread and from the first speculative thread to the second speculative thread~~, wherein the processor-readable medium includes at least one instruction to instruct the processor to trigger the invocation of the first and second speculative threads~~; and

an outstanding pre-computation slice counter to track for a set of loads of interest the number speculative threads that have been spawned relative to the number of instances of

any load of interest that have not yet been retired by the main thread and to decrement when the main thread retires the corresponding load of interest.

16. (Original) The processor of claim 15, further comprising a pending slice queue having entries to allocate a copy of at least one pre-computation slice when it is determined that a hardware context is unavailable for the copy of the pre-computation slice and to hold the pre-computation slice until a hardware context is available.

17. (Canceled).

18. (Currently Amended) The processor of claim 15, wherein the ~~further comprising an~~ outstanding pre-computation slice counter is further to ~~track for a set of loads of interest the number speculative threads that have been spawned relative to the number of instances of any load of interest that have not yet been retired by the main thread and to decrement en the main thread retires the corresponding load of interest, to~~ increment ~~the counter~~ when the corresponding pre-computation slice is spawned, and to force any speculative thread for which the entry in the counter is in a predetermined state to wait in the queue until the entry in the counter becomes a second predetermined state.

19. (Currently Amended) A machine-readable medium having machine-readable instructions stored thereon to instruct a processor to:

storing a main software thread in a first hardware context;

binding a first speculative thread to a second hardware context and a second speculative thread to a third hardware context;

dynamically invoke [[a]] the first speculative thread ~~from a~~ using the main thread ~~in a processor; and~~;

dynamically invoking the second speculative thread using the first speculative thread;

transferring live-in values from the main thread to the first speculative thread;

transferring live-in values from the first speculative thread to the second speculative thread;

tracking for a set of loads of interest a number speculative threads that have been spawned relative to a number of instances of any load of interest that have not yet been retired by the main thread; and

decrementing a counter when the main thread retires the corresponding load of interest

~~execute instructions comprising the speculative thread~~.

20. (Original) The machine-readable medium of claim 19 wherein the instructions are further to instruct the processor to:

allocate an entry in a queue for a copy of at least one pre-computation slice when it is determined that a hardware context is unavailable for the copy of the pre-computation slice; and

place the pre-computation slice in the queue until a hardware context is available.

21. (Canceled).

22. (Original) The machine-readable medium of claim 19 wherein the instructions are further to instruct the processor to:

identify a set of loads of interest;

construct pre-computation slices for each delinquent load of interest, wherein the pre-computation slice comprises a speculative thread that pre-computes an address accessed by a load of interest; and establish triggers to invoke the speculative thread.

23. (Original) The machine-readable medium of claim 22 wherein the instructions are further to instruct the processor to:

allocate an entry in a queue for a copy of at least one pre-computation slice when it is determined that a hardware context is unavailable for the copy of the pre-computation slice; and

place the pre-computation slice in the queue until a hardware context is available.

24. (Canceled).

25. (Currently Amended) The machine-readable medium of claim 23 wherein the instructions are further to instruct the processor to:

~~track for a subset of the set of loads of interest the number speculative threads that have been spawned relative to the number of instances of any load of interest that have not yet been retired by the main thread; and~~

increment a counter when the corresponding pre-computation slice is spawned; and

forcing any speculative thread for which the entry in the counter is in a predetermined state to wait in the queue until the entry in the counter becomes a second predetermined state.

Claims 26-31. (Canceled).